Intel® I/O Processors

Linux - Debian Installation Guide

June 2005

Order Number: 306507001US
June 2005

Contents

Figures

No Figures Used At This Time

Tables

Revision History

Date	Revision	Description
June 2005	001	Initial Release.

Introduction 1

The following instructions were validated using an IA host system running Fedora Core 3. The target is an IQ80331 booted in a Cyclone Microsystems* PCI Backplane. These instructions are also useful to those using a different setup, however there may be slight changes to accommodate different hosts or targets. Updates will be made to these instructions as changes are reported.

Host Setup 2

After the base install of the Linux distribution, ensure that the following packages are installed.

minicom:	This is a terminal emulation package for viewing the serial port console output of the Linux target system.
lrzsz:	X, Y, and Z modem support.
Native Development Tools:	For building the cross tool chain, a native version of GCC must be installed as well as various utilities needed by the build of the cross tool chain. During the Fedora installation, the option is given to install the Development tools packages. When chosen, adding these packages after the base installation, may require experimenting with the builds to ensure that all the dependencies are installed.
NFS server:	The target system may have its root file system mounted over NFS. This is not necessary when the target uses a Flash disk or a locally attached hard drive.
Remote login client:	This can be either a telnet or SSH client, depending on plans for remote connectivity. (SSH is encrypted and much more secure than telnet; however, the SSH daemon is a bigger package.) This may not be necessary when serial console access is sufficient for required needs.
TFTP Server:	RedBoot uses TFTP to load the kernel from the host system. This is not necessary when using XModem for kernel downloads. Note: It is highly recommended to install and use the TFTP server. A kernel image takes many times longer to load over serial than over ethernet.
DHCPD Server:	This provides BOOTP information for the target. This is not necessary when assigning static IP addresses.

On my Fedora Core 3 system, I have the following packages installed to meet these requirements:

- coreutils-5.2.1-31
- dhcp-3.0.1-11
- gawk-3.1.3-9
- gcc-3.4.2-6.fc3
- gcc-c++-3.4.2-6.fc3
- grep-2.5.1-31.2
- libgcc-3.4.2-6.fc3
- lrzsz-0.12.20-19
- make-3.80-5
- minicom-2.00.0-19
- openssh-3.9p1-7
- openssh-clients-3.9p1-7
- openssh-server-3.9p1-7
- nfs-utils-1.0.6-44
- sed-4.1.2-4
- system-config-nfs-1.2.8-1
- telnet-0.17-30
- tftp-0.39-1
- tftp-server-0.39-1

Note: When running a firewall on the host system, ensure that connections are accepted from the target board. See the firewall or system documentation for more details.

2.1 Minicom Setup

Follow these steps to configure the minicom terminal emulator:

1. Log in as root
2. Start minicom setup (“minicom -s”)
3. Select [Serial port setup]
   a. Change Serial Device to the appropriate com port (/dev/ttyS0 is COM1, /dev/ttyS1 is COM2 etc.)
   b. Change Bps/Par/Bits to 115200 8N1, this is the default setting for all the Customer Reference Board Evaluation Platforms.
   c. Change all flow control settings to No/Off
4. Select [Modem and dialing parameter setup]
   a. Change Init string to “^M”
   b. Change Reset string to “^M”
5. Select [Save setup as dfl] to save setup as default
6. Select [Exit] to exit setup and enter minicom
7. Log out as root

2.2 Host Networking Setup

These instructions assume there are two ethernet controllers and are using a private subnet of 192.168.0.0 for the Intel XScale® microarchitecture-Linux target. Normal disclaimers about security apply.

Launch the system-config-network utility on the Fedora host system and configure the secondary ethernet controller with an IP of 192.168.0.1 and a subnet mask of 255.255.255.0.

When providing full internet access to the Intel XScale® microarchitecture-Linux target, it is possible to configure the IA Linux host to act as a NAT router, connecting the 192.168.0.0 subnet to the primary network. Instructions to do this vary depending on the distribution. Please consult the appropriate vendor documentation for this.

2.3 DHCPD Server Setup

Create or edit the file /etc/dhcpd.conf. The actual dhcpd.conf file may be different than this depending on the needs, however, this is a reasonable starting point.

```
deny unknown-clients;
ddns-update-style ad-hoc;
allow bootp;

subnet 192.168.0.0 netmask 255.255.255.0 {
}

group {
    option broadcast-address 192.168.0.255;
    option domain-name "xscale-iop.net";
    option routers 192.168.0.1;
    option subnet-mask 255.255.255.0;

host myiop.xscale.net {
    # put your MAC address here and replace 0's
    hardware ethernet 00:00:00:00:00:00;
    fixed-address  192.168.0.100;
    option host-name "myiop";
    option root-path "/exports/fs.xscale";
    }
}
```

Ensure that the /var/lib/dhcp/dhcpd.leases file exists. When not, create it with the command “touch /var/lib/dhcp/dhcpd.leases”.

Start the dhcpd server. On Fedora or Mandrake systems, issue the command “/sbin/service dhcpd start”. Other platforms may invoke the rc init script for dhcpd as “/etc/init.d/dhcpd start”.

For Fedora and Mandrake host systems, ensure that the DHCP server runs automatically at boot by issuing the command “/sbin/chkconfig dhcpd on”.

Note: When using BOOTP or DHCP to assign dynamic IP addresses, use the 'fconfig' command in RedBoot to configure the board to query for an IP address. See the RedBoot User's Manual

2.4 NFS Server Setup

When Intel XScale® microarchitecture-Linux target is using NFS for its root file system, it is necessary to export the appropriate directory via the NFS Server. These instructions assume the NFS mounted root file system is located on the server as "/exports/fs.xscale".

Edit the file "/etc/exports" and add the following line:

```
/exports/fs.xscale *(rw,no_root_squash,sync)
```

Create the NFS shared directory:

```
# mkdir /exports/fs.xscale
```

Start NFS server as follows on a Fedora or Mandrake server:

```
# /sbin/service portmap start
# /sbin/service nfslock start
# /sbin/service nfs start
```

Ensure that the NFS server will be automatically run at boot time as follows on a Fedora or Mandrake server:

```
# /sbin/chkconfig portmap on
# /sbin/chkconfig nfslock on
# /sbin/chkconfig nfs on
```

When the NFS server was started before editing the "/etc/exports" file, it is necessary to update the export tables as follows:

```
# exportfs -rav
```

2.5 TFTP Server Setup

This example is based on a Fedora system. Other systems may provide similar mechanisms for this. Consult the distributions documentation for further information.

Enable the TFTP server:

```
# /sbin/chkconfig tftp on
```

Edit the "/etc/xinetd.d/tftp" file and ensure that the following settings are present. These may be default so it may not be necessary to modify anything:

```
disable   = no
user      = root
server_args= -s /tftpboot
```

Create the TFTP Root directory when it does not exist.

```
# mkdir /tftpboot
```

Restart the xinetd daemon:

```
# /sbin/service xinetd restart
```

Target Firmware Setup 3

Connect the ethernet port of the Intel XScale® microarchitecture CRB to the server secondary ethernet controller via a crossover cable or connect them both to a GbE hub. Connect the serial port of the CRB to the serial port of the server. Running “minicom” should allow viewing of the RedBoot console output from the board.

Ensure that the latest RedBoot is installed on the CRB. Currently the 20050321 IOP RedBoot release is the latest and binaries are available from http://developer.intel.com/design/iio/devkits/softwaresupport.htm.

Should the board not have a valid RedBoot image installed, consult the board documentation for instructions on Flashing via FRU or JTAG.

Assuming the board has a bootable RedBoot image installed, an update can be done via commands from RedBoot. Parameters in the instructions below will vary from board to board. Table 1 gives the values for each of the SCD CRBs.

Table 1. RedBoot Parameter Values for CRBs

{board}	{img_len}	{flash-addr-rom-img}	{ram-addr-rom-img}	{flash-addr-ram-img}	{ram-addr-ram-img}
iq8033x	0x40000	0xc0000000	0x100000	0xc0040000	0x20000
iq80321	0x40000	0xf0000000	0x100000	0xf0040000	0x20000
ep80219	0x40000	0xf0000000	0x100000	0xf0040000	0x20000
iq31244	0x40000	0xf0000000	0x100000	0xf0040000	0x20000
iq80315	0x100000	0x40000000	0x400000	0x40100000	0x200000

To download via XModem:

```
RedBoot> load -m xmodem
```

Note: <download via xmodem {board}-ram\install\bin\redboot.srec>

```
RedBoot> fis unlock -f {flash-addr-ram-img} -l {img_len}
RedBoot> fis create RedBoot[backup] -f {flash-addr-ram-img}
    -b {ram-addr-ram-img} -r {ram-addr-ram-img} -l {img_len}
RedBoot> fis lock -f {flash-addr-ram-img} -l {img_len}
RedBoot> fis load RedBoot[backup]
RedBoot> go
RedBoot> load -m xmodem -b {ram-addr-rom-img}
```

Note: <download via xmodem {board}-rom\install\bin\redboot.srec>

```
RedBoot> fis unlock -f {flash-addr-rom-img} -l {img_len}
RedBoot> fis create RedBoot -f {flash-addr-rom-img}
    -b {ram-addr-rom-img} -l {img_len} -s {img_len}
RedBoot> fis lock -f {flash-addr-rom-img} -l {img_len}
RedBoot> reset
```

When preferred, use the TFTP server to download the images. This is much faster that XModem. Copy the {board}-rom\install\bin\redboot.srec and {board}-ram\install\bin\redboot.srec to the TFTP server under unique names such as redboot-rom.srec and redboot-ram.srec.

Then issue the following command sequence:

```
RedBoot> load -m tftp redboot-ram.srec
RedBoot> fis unlock -f {flash-addr-ram-img} -l {img_len}
RedBoot> fis create RedBoot[backup] -f {flash-addr-ram-img}
    -b {ram-addr-ram-img} -r {ram-addr-ram-img} -l {img_len}
RedBoot> fis lock -f {flash-addr-ram-img} -l {img_len}
RedBoot> fis load RedBoot[backup]
RedBoot> go
RedBoot> load -m tftp -b {ram-addr-rom-img} redboot-rom.srec
RedBoot> fis unlock -f {flash-addr-rom-img} -l {img_len}
RedBoot> fis create RedBoot -f {flash-addr-rom-img} -
    b {ram-addr-rom-img} -l {img_len} -s {img_len}
RedBoot> fis lock -f {flash-addr-rom-img} -l {img_len}
RedBoot> reset
```

Cross Toolchain 4

The next step in building a Linux system for running on Intel XScale® microarchitecture is to generate a cross-toolchain for building the kernel and applications from the IA Linux host. Currently, we use a toolchain built using the crosstool build scripts (http://kegel.com/crosstool). We currently build for the armv5l-linux target.

After the latest version of the script is downloaded a couple changes need to be made:

1. Edit the *arm-xscale.dat* file to change the following line:

   TARGET=arm-xscale-linux-gnu

   To

   TARGET=armv5l-linux

2. Edit the *demo-arm-xscale.sh* to select which versions of gcc and glibc are to be used. Comment out or remove all of the statements starting with "eval `cat" and replace them with the following statement:

   ```
   eval `cat arm-xscale.dat gcc-3.4.3-glibc-2.3.5.dat` sh all.sh --notest
   ```

Change the RESULTS_TOP variable to the directory where the tools are to be installed (the default is "/opt/crosstool")

After the changes are made the *demo-arm-xscale.sh* script needs to be run. Next, add the /opt/crosstool/bin (or the path where the tools were built) to your PATH variable.

Kernel 5

The next step is to build an appropriate kernel. IOP Kernel patches are available at http://sourceforge.net/projects/xscaleiop/. Grab the latest 2.6 kernel patch, and the corresponding kernel source tarball from http://www.kernel.org/.

The following commands will configure the 2.6.10 IOP kernel for an IQ80331 platform.

$ mkdir ~/work

$ cd ~/work

$ wget http://aleron.dl.sourceforge.net/sourceforge/xscaleiop/patch-2.6.10-iop1.bz2

$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.10.tar.bz2

$ tar xjvf linux-2.6.10.tar.bz2

$ cd linux-2.6.10/

$ bzcat ../patch-2.6.10-iop1.bz2 | patch -p1

$ make iq80331_defconfig

– substitute the board name here when it is not the IQ80331

When planning on installing Debian-ARM, it is necessary to add INITRD support to the kernel.

Note: INITRD support is only necessary for running the installer or when creating INITRD images. The default setup does not require INITRD support in the kernel so once the base installation is complete, it is possible to remove INITRD support from the kernel

$ make menuconfig

– Scroll down to “Block Devices” and press Return.

– Scroll down to “Initial RAM disk (initrd) support” and press the space bar until it is selected with an '*'.

– At this point it is necessary to enable other drivers that might be needed.

– Exit all the way out of the menuconfig and save the new kernel configuration.

Now build the kernel

$ make zImage

With a multiprocessor system, increase the speed of the kernel build by allowing make to issue multiple jobs at one. A 4-processor system has run;

$ make -j 4 zImage

Now copy the kernel to the TFTP directory

$ cp arch/arm/boot/zImage /tftpboot/zImage.iq80331

Now test that the kernel boots. It is OK for there to be no root file system at this point. This is only a trial test of the kernel.

Table 2 gives the parameter values for each of the SCD CRBs.

Table 2. Kernel Parameter Values for CRBs

{board}	{kernel-load-addr}	{kernel-run-addr}	{ramdisk-load-addr}	{ramdisk-run-addr}
iq8033x	0x01008000	0x01008000	0x00800000	0x00800000
iq80321	0x01008000	0xa1008000	0x00800000	0x00800000
ep80219	0x01008000	0xa1008000	0x00800000	0x00800000
iq31244	0x01008000	0xa1008000	0x00800000	0x00800000
iq80315	0x01008000	0x01008000	0x00800000	0x00800000

```
+Ethernet eth0: MAC address 00:0e:0c:52:98:20
IP: 10.0.1.8/255.255.255.0, Gateway: 10.0.1.9
Default server: 10.0.1.9, DNS server IP: 0.0.0.0

RedBoot bootstrap and debug environment [ROM]
    Intel IOP RedBoot release
    version 2.0-IOP-RedBoot
    built 08:01:31; December 21, 2004

Platform: IQ80331 (Intel XScale® microarchitecture)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x08000000, 0x0001af78-0x07fd1000 available
FLASH: 0xc0000000 - 0xc0800000, 64 blocks of 0x00020000 bytes each.
RedBoot> load -r -v -b {kernel-load-addr} zImage.iq80331
Using default protocol (TFTP)
-
Raw file loaded 0x.....-0x....., assumed entry at 0x.....
RedBoot> exec {kernel-run-addr}
Using base address 0x..... and length 0x.....
The boot tags are located at 0x00000100
Booting the kernel...
Uncompressing Linux.....
Linux version 2.6.10-iop1 (user@labcomp) (gcc version 3.4.2) #1 Wed J5
CPU: XScale-IOP8033x Family [69054097] revision 7 (ARMv5TE)
CPU: D VIVT undefined 5 cache
CPU: I cache: 32768 bytes, associativity 32, 32 byte lines, 32 sets
CPU: D cache: 32768 bytes, associativity 32, 32 byte lines, 32 sets
Machine: Intel IQ80331
Memory policy: ECC disabled, Data cache writealloc
```

```
Built 1 zonelists
Kernel command line: ip=boot root=nfs console=ttyS0,115200 cachepolicy=writeallc
PID hash table entries: 1024 (order: 10, 16384 bytes)
Console: colour dummy device 80x30
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 128MB = 128MB total
Memory: 126720KB available (2289K code, 445K data, 320K init)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
CPU: Testing write buffer coherency: ok
NET: Registered protocol family 16
PCI: bus0: Fast back to back transfers disabled
SCSI subsystem initialized
Intel IOP3XX DMA Copyright(c) 2004 Intel Corporation
NetWinder Floating Point Emulator V0.97 (double precision)
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
SGI XFS with ACLs, security attributes, no debug enabled
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
ttyS0 at MMIO 0xfffff700 (irq = 51) is a XScale
ttyS1 at MMIO 0xfffff740 (irq = 52) is a XScale
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
Intel(R) PRO/1000 Network Driver - version 5.5.4-k2-NAPI
Copyright (c) 1999-2004 Intel Corporation.
e1000: eth0: e1000_probe: Intel(R) PRO/1000 Network Connection
elevator: using anticipatory as default io scheduler
physmap flash device: 800000 at c0000000
phys_mapped_flash: Found 1 x16 devices at 0x0 in 8-bit bank
Intel/Sharp Extended Query Table at 0x0031
.......
```

At this point we can be fairly confident that our toolchain and kernel are sane. Now we must install a distribution.

Distributions 6

There are numerous choices for the root file system or distribution.

Some distributions, such as Debian, are desktop-type distributions that have been ported to the ARM processor family. These distributions will give much flexibility in terms of application availability and an active developer community. These distributions will typically take a reasonably large amount of disk space and will likely be inappropriate for installation into a flash disk. Applications are typically distributed as precompiled binaries with little configuration of optimization levels, etc.

Some distributions, such as the ucLinux distribution, are geared toward embedded projects with limited storage space. These distributions can normally be tailored for an custom installation, and will generally produce the smallest footprint file systems. They generally have a more limited developer community, so they may require more custom work for a given task. The applications here are typically compiled from scratch, and a lot of times they are cross-built from the IA Linux host. This gives fine-grained control over optimization levels, etc.

6.1 Installing Debian

See http://www.debian.org for more information on the Debian distribution.

There are three main branches of Debian: stable (currently codenamed Woody), testing (currently codenamed Sarge), and unstable (currently codenamed Sid). (See http://www.debian.org/releases/ for more information).

Installers for both Testing and Unstable exist; however, there were issues installing directly. When interested in either of these releases, install the stable release first (see Section 6.1.1) and then use the "apt-get" utility to upgrade to the new version of the distribution. (see Section 6.1.5). For most users the Testing branch is sufficiently stable. In addition, the Testing branch offers higher performance because it is built with GCC 3.3. The Stable branch has been tested much more thoroughly but is based on GCC 2.95, which has lower performance.

6.1.1 Installing Debian Stable on a NFS Root File System

First, ensure that the "/exports/fs.xscale" directory exists and is empty, and then make sure that the NFS server is exporting the correct directory.

```
# rm -rf /exports/fs.xscale     (when not caringabout what is there currently)
# mkdir /exports/fs.xscale
# exportfs -rav
```

Now, download the installer disk image and copy it into the TFTP server root directory.

```
# wget http://http.us.debian.org/debian/dists/woody/main/disks-arm/current/
netwinder/images-1.44/root.bin
# mv root.bin /tftpboot/initrd.gz
```

Now, on the RedBoot console, issue the following commands to launch the installer:

```
RedBoot> load -r -v -b {ramdisk-load-addr} /initrd.gz
RedBoot> load -r -v -b {kernel-load-addr} /zImage.iq80331
RedBoot> exec {kernel-run-addr} -r {ramdisk-run-addr}
    -s 0x400000 -c "ip=bootp root=/dev/ram0 \
    initrd={ramdisk-run-addr},
    4M console=ttyS0, 115200 cachepolicy=writealloc"
```

After the kernel boots, the Splash Screen of the Debian Installer is viewable. Execute the following steps to get a base installation of Debian Linux.

1. Press return to get to the "`Installation Main Menu`".
2. Scroll to "`Mount a Previously-Initialized Partition`" and press return.
3. Enter the NFS path and press return: "`192.168.0.1:/exports/fs.xscale`"
4. Enter "`yes`" to mount this NFS share as the root file system:
   a. This may take some time to complete. Be patient.
      When problems are suspect, check the log files on the server.
5. Scroll to "`Execute a shell`" and press return.

6. There is an issue with the dpkg program that is part of the Debian installer. It fails to create lock files correctly when they are located on an NFS drive. The particular lock files are located in /target/var/lib/dpkg, so we will address this issue by creating a RAM disk and mounting it as /target/var/lib. Execute the following commands:

   ```
   # mke2fs /dev/ram1
   # mkdir -p /target/var/lib
   # mount -t ext2 /dev/ram1 /target/var/lib
   # exit
   ```

7. Scroll to "`Configure the hostname`" and press return.
8. Enter a name for this system or accept the default and press return.
9. Scroll to "`Install the Base System`" and press return.
10. Select "`Network`" to install the base system and press return.
11. Accept the default download URL or enter a new one.
12. When a proxy server is required, enter the IP address and port number of the proxy server in the next dialog box. Note: Name resolution is not enabled at this step of the installation, so an IP address (ex., 12.34.56.78) is required, a domain name (ex., proxy.domain.com) will NOT work.
13. This takes some time since it downloads the base packages and installs them, when the base installation is complete, the "`Installation Main Menu`" reappears.
14. Before Rebooting, we must ensure that the files that are in the RAM disk are saved to the NFS share. Scroll to "Execute a Shell", press return, and enter the following commands:

    ```
    # cd /target/var
    # tar cvf lib.tar lib
    # exit
    ```

    On the Linux host, issue the following commands to untar the files from the RAM Disk.

    ```
    # cd /exports/fs.xscale/var
    # tar xvf lib.tar
    # rm -f lib.tar
    ```

15. Scroll to "`Reboot the system`" and press return.
16. Select "`<Yes>`" and press return to reboot.
17. Remove the" `/dev/ram1`" line from the "/exports/fs.xscale/etc/fstab" file on the host system.
18. Since we are using kernel-level autoconfiguration, the networking scripts on the target do not install the "/etc/resolv.conf" file which allows us to do name resolution properly. On the IA Linux host, Create the "/exports/fs.xscale/etc/resolv.conf" file that looks like the following:

    nameserver "`192.168.0.1`", make sure that the proper IP address of a nameserver on the network is included.

At this point the Base Installation is complete and the board should reboot to the RedBoot prompt. When wanting to install to a locally attached hard disk, a similar procedure can be used with changes as follows:

- In step 2 above, the installer should detect the hard disk and prompt for a partition and format the disk rather than having to enter NFS information.
- Additional installation steps may be useful depending on the needs; swap disks, additional partitions, etc.
- The workarounds in steps 5, 6, and 15 are not necessary.
- It is necessary to create the resolv.conf file on the disk.

To boot the new Debian NFS root based system, issue the following commands at the RedBoot prompt:

```
RedBoot> lo -r -v -b {kernel-load-addr} /zImage.iq80331
RedBoot> exec {kernel-run-addr}
```

The default Kernel command line parameters are setup for a Root NFS based system, with the NFS root directory name being taken from the DHCP server. The kernel command line parameters can be modified from the RedBoot prompt. The following example will use a locally attached disk as the Root File system. When the disk is not sda1, it is necessary to modify this as appropriate for the setup.

```
RedBoot> lo -r -v -b {kernel-load-addr} /zImage.iq80331
RedBoot> exec {kernel-run-addr} -c "ip=bootp root=/dev/sda1
    console=ttyS0,115200 cachepolicy=writealloc"
```

6.1.2 Initial Debian System Configuration

On the first boot of the new Debian installation, the base-config application will be run. This will do more configuration of the system. Please consult the Debian home page for help with this process. I generally skip running the "`tasksel`" and "`dselect`" processes until I have tweaked the configuration of my system as discussed next.

When running Linux on any of the IOP HBA-type CRBs (i.e., IQ80321, IQ80331, etc), they will not have a real-time-clock (RTC) on board. As such, they will not retain the time across boots. To address this, we can install either rdate or ntp. A side effect of this is that for every login, there is a prompted to change the password. This is because password aging is enabled be default and the date is wrong, thus confusing the system.

6.1.3 Maintaining System Time With Rdate

To enable the IA Linux host as an rdate server, issue the following commands on the server as root:

```
# /sbin/chkconfig time on
# /sbin/service xinetd restart
```

Then on the Intel XScale® microarchitecture-Linux target, issue the following commands to download and install the rdate package. This only needs to be done once.

```
# apt-get install rdate
```

To manually update the time on the target:

```
# rdate 192.168.0.1
```

Add the following to the '/etc/rc.local' or create it when it does not exist:

```
rdate -s 192.168.0.1
```

and ensure that it is executable and is called on each boot:

```
# chmod 755 /etc/rc.local
# cd /etc/rc2.d; ln -s ../rc.local ./S99local
# cd /etc/rc3.d; ln -s ../rc.local ./S99local
# cd /etc/rc4.d; ln -s ../rc.local ./S99local
# cd /etc/rc5.d; ln -s ../rc.local ./S99local
```

6.1.4 Maintaining System Time With NTP and NTPDATE

Issue the following commands to install and configure ntpdate to run:

```
# apt-get install ntpdate
```

To install and run the NTP daemon, issue the following commands:

```
# apt-get install ntp
```

Now we can rerun tasksel and dselect applications to select other programs and finish our installation.

6.1.5 Upgrading to Debian Testing

To upgrade the Debian installation to testing:

1. First edit the file “/etc/apt/sources.list”.
2. Replace the word “stable” with “testing” wherever it appears.
3. Then run the following:

   ```
   # apt-get update
   # apt-get dist-upgrade
   ```